LES BASES DE REACTIVE X

QUESTION TRÈS TRÈS IMPORTANTE...

VOUS AVEZ UN IPHONE
OU UN ENDRÖIDEUH ?


RX, C'EST QUE DU LOVE !

C'EST POUR QUOI FAIRE, ET SUR QUOI ?

Les français veulent savoir !

"ReactiveX is everywhere, and it's meant for everything."

- **Cross-Platform** : a lot of languages
- **Frontend** : UI, API responses, web, mobile, ...
- **Backend** : async, concurrency, ...

“ReactiveX is more than an API, it's an idea and a breakthrough in programming.

It has inspired several other APIs, frameworks, and even programming languages.”

RX, KÉSSESSÉ ?

Montrez-nous ce qu'il y a sous le capot.

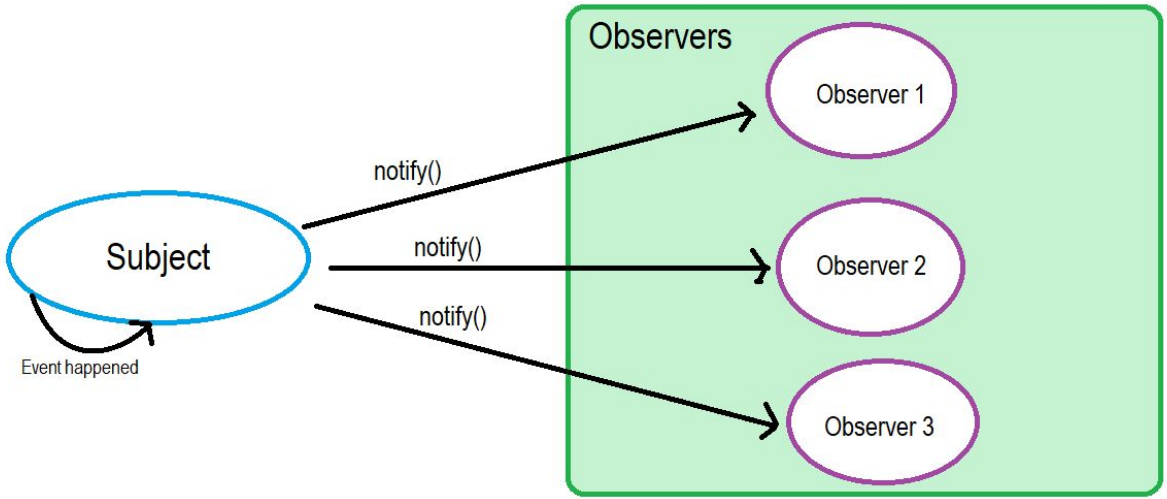
“ReactiveX is a combination of the best ideas from

- the Observer pattern,
- the Iterator pattern,
- and functional programming.”

OBSERVER ?

On s'abonne, et on écoute sagement.

OBSERVER PATTERN



Rx : trois événements pour notifier

next

- émis avec de nouvelles données.
- peut être envoyé 0 à n fois.

RX : TROIS ÉVÉNEMENTS POUR NOTIFIER

completed

- marque la fin du traitement.
- peut être envoyé 0 ou 1 fois.
- le flux se termine après un completed !

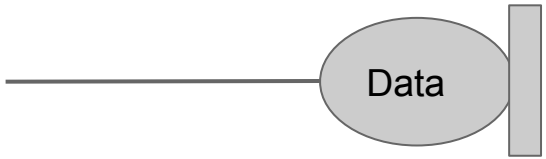
RX : TROIS ÉVÉNEMENTS POUR NOTIFIER

error

- une erreur lors du traitement.
- peut être envoyé 0 ou 1 fois.
- le flux se termine après un error !

EXEMPLES D'UNE REQUÊTE HTTP

next(data) + completed



error(infos)

```
Api.requestSomething()
    .subscribe(
        onNext: { (data: Data) in
            // use data, it's typed and non-optional !
        }, onError: { (error: Error) in
            // use error, it's typed and non-optional !
        }, onCompleted: {
            // do something when it's completed.
        })
```

ITERATOR ?

On parcourt, mais pas que ...

Iterator pattern

- Tout est collection et séquence.

Flux => liste d'événements

On y applique des opérations

ITERATOR PATTERN

- L'Observable est un container.

On n'expose que la manière de l'itérer (next, completed, error).

```
Api.requestSomething()
    .subscribe(
        onNext: { (data: Data) in
            // use data, it's typed and non-optional !
        }, onError: { (error: Error) in
            // use error, it's typed and non-optional !
        }, onCompleted: {
            // do something when it's completed.
        })
```

FUNCTIONNAL PROGRAMMING ?

Fonctionnellement parlant, c'est plutôt fonctionnel.

Functional programming

Pas d'effet de bord :

- immutable
- pas de machine à états complexe

FUNCTIONAL PROGRAMMING

Pas d'effet de bord :

- c'est un emboîtement de fonctions

Et il y a tout un tas d'opérateurs !

DES OPÉRATEURS ?

Tout ce dont vous avez toujours rêvé.

L'opérateur de Reactive X

Un opérateur est une fonction que l'on applique sur un Observable.

On va transformer, influencer, modifier le comportement et le contenu du flux.

Chaining operators

Un opérateur retourne un Observable, avec la fonction appliquée.

On peut chainer autant d'opérateurs que l'on souhaite.

```
Api.requestSomething()
    .retry(3)
    .map { (data: Data) -> Account in
        parseDataIntoAccount(data)
    }
    .filter { $0.age > 18 }
    .subscribe(
        onNext: { (account: Account) in
            print(account)
    })
```

UNE BELLE PROMESSE

Il y a un opérateur pour tout ce que vous souhaitez faire.

Et si vous ne trouvez pas, c'est que votre traitement est mal découpé.

Aperçu d'un aperçu

Create, Defer, Empty/Never/Throw, From, Interval, Just, Range, Repeat, Start, Timer… Buffer, FlatMap, GroupBy, Map, Scan, Window… Debounce, Distinct, ElementAt, Filter, First, Last, Skip, SkipLast, Take, TakeLast… And/Then/When, CombineLatest, Join, Merge, StartWith, Switch, Zip… Catch, Retry… Delay, Do, Materialize/Dematerialize, ObserveOn, Serialize, Subscribe, SubscribeOn, TimeInterval, Timeout, Timestamp, Using… All, Amb, Contains, DefaultIfEmpty, SequenceEqual, SkipUntil, SkipWhile, TakeUntil, TakeWhile… Average, Concat, Count, Max, Min, Reduce, Sum… Connect, Publish, RefCount, Replay…

Les principaux opérateurs ?

Une petite sélection des plus utilisés.

Quelques incontournables

- map, flatMap
- filter
- debounce
- distinctUntilChanged
- observeOn, subscribeOn
- debug
- catchError, catchErrorJustReturn
- zip, combineLatest
- create, just, from

COMMENT ON S'Y RETROUVE DANS TOUT ÇA ?

Si seulement on pouvait avoir une aide graphique...

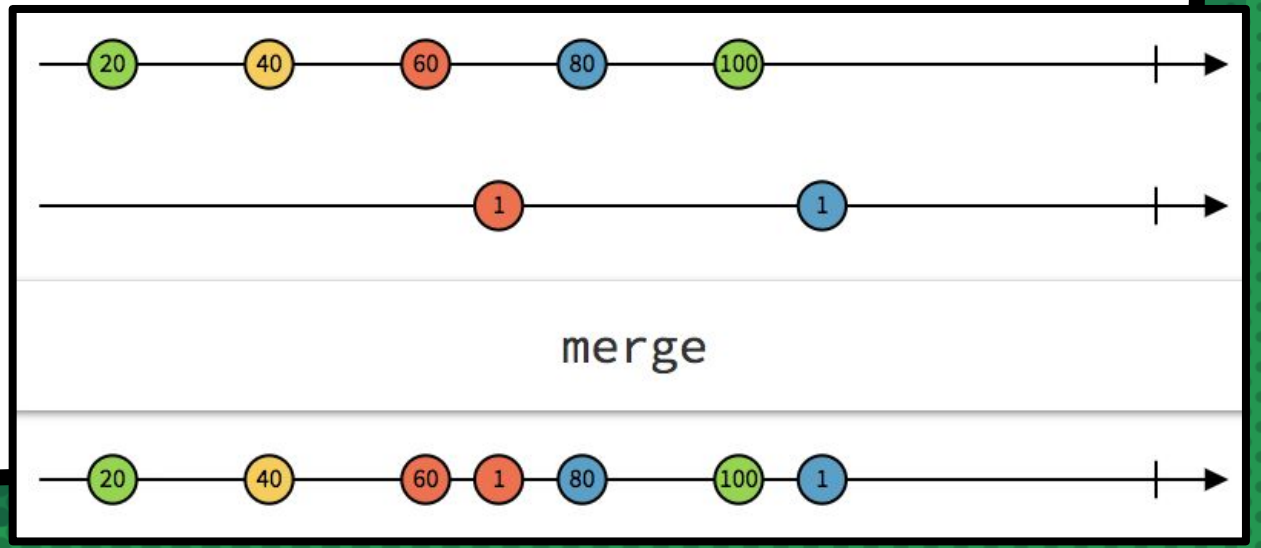
ReactiveX.io

Site officiel

- Grande documentation.
- Explications des concepts.
- Présentation des opérateurs.

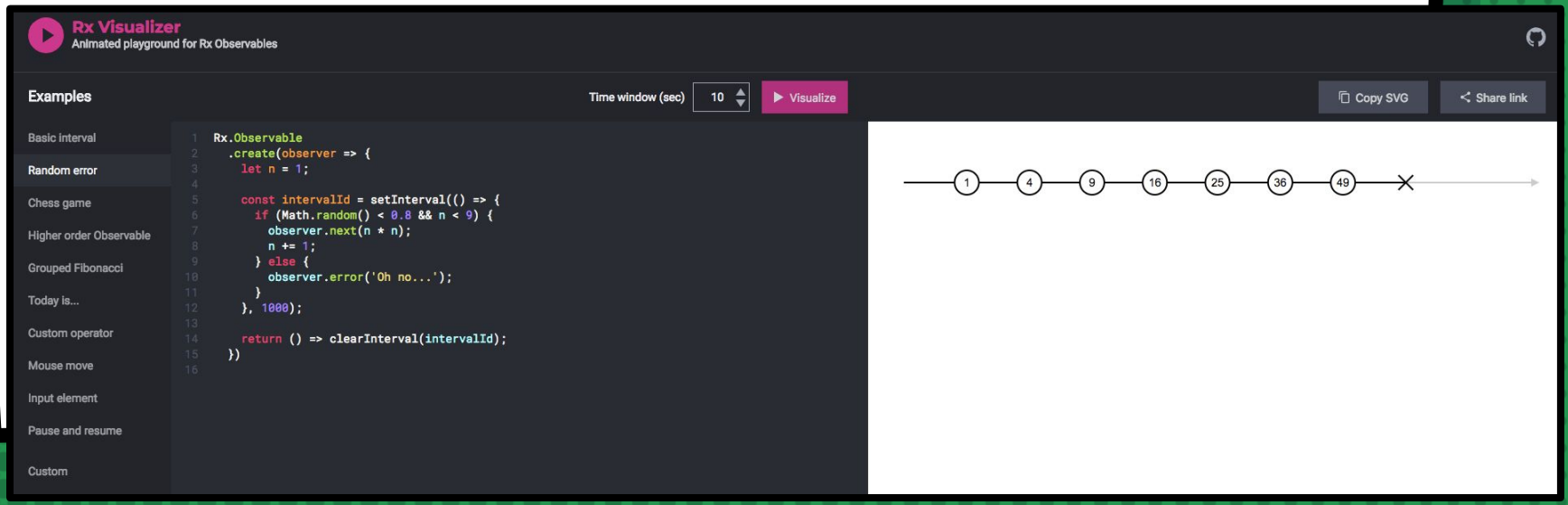
RXMARBLES.COM

Diagrammes interactifs



RXVIZ.COM

Bac à sable pour tester et visualiser



Des exemples de cas d'utilisation ?

J'utilise, tu utilises, nous utilisations.

Binding UI et Data

- MVVM – ViewModel
- Une vue toujours à jour, dès que la donnée change

Combiner des traitements asynchrones

- Traitements async successifs
- Groupement de traitements

Utiliser la puissance des opérateurs

- Bon découpage
- Meilleure lecture
- Ne réinvente pas la roue

DOJO !

ON FAIT TOURNER LE CLAVIER

TOUTES LES 5 MINUTES

THANKS!

Des questions ? Des remarques ?

Retrouvez-moi sur Twitter @Yupjoo